**Proyecto Final**

**Aporte Redis-Kakfa**

**Elaborado por:**

Lisbeth Callata Churata

**ÍNDICE**

[**I.** **INTRODUCCIÓN** 3](#_Toc184181743)

[**II.** **OBJETIVO:** 3](#_Toc184181744)

[**III.** **MARCO TEORICO** 3](#_Toc184181745)

[3.1 Redis 3](#_Toc184181746)

[3.2 Kafka 3](#_Toc184181747)

[**IV.** **CASO DE USO** 4](#_Toc184181748)

[4.1 Casos de uso reales de Redis: 4](#_Toc184181749)

[4.2 Casos de uso reales de Kafka: 5](#_Toc184181750)

[**V.** **APLICACIÓN EN EL PROYECTO DE MICROSERVICIOS** 6](#_Toc184181751)

[5.1 Casos de uso real de Redis: 6](#_Toc184181752)

[5.2 Casos de uso real de Kafka: 6](#_Toc184181753)

[**VI.** **BENEFICIOS** 7](#_Toc184181754)

[**VII.** **CONCLUSIONES** 7](#_Toc184181755)

1. **INTRODUCCIÓN**

En un sistema de microservicios, cada componente debe ser escalable, eficiente y resiliente para garantizar un alto rendimiento y una fácil integración. Redis y Kafka son dos herramientas potentes que pueden mejorar significativamente el rendimiento y la escalabilidad de tu arquitectura. Redis se destaca como una base de datos en memoria rápida y eficiente para caché, gestión de sesiones y almacenamiento temporal. Kafka, por su parte, es una plataforma distribuida de mensajería orientada a eventos, ideal para procesar grandes volúmenes de datos en tiempo real y facilitar la comunicación entre servicios de manera asíncrona. En este informe se señalará los principales aportes de ambas herramientas en casos reales y aplicados en el sistema bancario desarrollado.

1. **OBJETIVO:**
2. Explicar cómo Redis y Kafka pueden ser aplicados en un sistema de microservicios bancario, detallando sus casos de uso, ventajas y su integración dentro del contexto de los microservicios.
3. Explorar casos de uso reales de Redis y Kafka, para entender cómo estas herramientas pueden ser aplicadas específicamente en un entorno real.
4. **MARCO TEORICO**
   1. **Redis**

Redis es una base de datos en memoria que permite almacenar y acceder a datos con gran rapidez. Redis es comúnmente utilizado para caché, gestión de sesiones y almacenamiento temporal de datos. Algunas características clave de Redis incluyen:

* Alto rendimiento: Al estar basado en memoria, las operaciones en Redis son extremadamente rápidas.
* Persistencia opcional: Si bien Redis es una base de datos en memoria, ofrece mecanismos de persistencia como RDB snapshots y AOF (Append Only File).
* Estructuras de datos avanzadas: Redis ofrece una variedad de estructuras de datos como listas, conjuntos, mapas hash, entre otros.
  1. **Kafka**

Kafka es una plataforma de mensajería distribuida y un sistema de procesamiento de flujos de datos en tiempo real. Está diseñado para gestionar grandes volúmenes de eventos y datos en tiempo real, ofreciendo alta disponibilidad, escalabilidad y durabilidad. Las características principales de Kafka incluyen:

* Alto rendimiento y escalabilidad: Kafka puede manejar millones de eventos por segundo, lo que lo convierte en una excelente opción para sistemas distribuidos de alto rendimiento.
* Durabilidad y resiliencia: Los mensajes en Kafka pueden persistirse en discos para garantizar que no se pierdan, incluso en caso de fallos.
* Desacoplamiento de servicios: Kafka actúa como un intermediario entre microservicios, desacoplando las dependencias directas y facilitando la comunicación asíncrona.

1. **CASO DE USO**
2. **Casos de uso reales de Redis:**
3. **Caché de resultados**:

En el sistema bancario, se puede usar Redis para almacenar en caché resultados frecuentes, como el saldo de las cuentas o la información de los clientes. Esto reduce la carga en la base de datos principal (MongoDB) y mejora el rendimiento, especialmente cuando se consultan los mismos datos repetidamente.

**Ejemplo**:

Cada vez que un usuario consulta el saldo de su cuenta, puedes verificar primero en Redis si el dato está disponible en caché antes de consultar la base de datos MongoDB. Si el saldo está en la caché, se devuelve rápidamente. Si no, se consulta la base de datos y luego se almacena el resultado en Redis para futuras consultas.

1. **Gestión de sesiones de usuario**:

Si el sistema requiere que los usuarios inicien sesión, puedes usar Redis para gestionar sesiones. Redis permite almacenar y acceder rápidamente a los datos de sesión del usuario.

**Ejemplo**:

Después de que un cliente se loguea en el sistema, se almacena un token de sesión en Redis. Este token puede ser utilizado en futuras solicitudes para autenticar al usuario sin tener que consultar la base de datos de clientes cada vez.

1. **Contadores y límites**:

Redis es ideal para mantener contadores en tiempo real, como la cantidad de transacciones realizadas por un cliente, o el número de intentos fallidos de inicio de sesión. Esto puede ser útil para implementar medidas de seguridad, como bloquear una cuenta después de varios intentos fallidos.

**Ejemplo**:

Si un usuario intenta realizar varias transacciones en un corto período de tiempo, Redis puede mantener un contador para asegurarse de que no se sobrepasen los límites establecidos.

1. **Casos de uso reales de Kafka:**
2. **Eventos de transacciones bancarias**:

Kafka puede utilizarse para enviar eventos cuando se realizan transacciones bancarias. Cada vez que un cliente realice un depósito, retiro o transferencia, un mensaje puede ser enviado a un tema de Kafka. Otros sistemas (por ejemplo, un sistema de auditoría) pueden consumir estos mensajes en tiempo real para realizar auditorías o generar reportes.

**Ejemplo**:

Cada vez que se realice una transacción de depósito, un mensaje como "Depósito realizado: $1000 a cuenta X" puede ser enviado a un topic de Kafka. Un sistema de auditoría podría consumir ese mensaje y registrar la transacción.

1. **Procesamiento en tiempo real de pagos**:

En un escenario en el que las transacciones sean muy rápidas y en tiempo real, Kafka puede ser utilizado para procesar eventos como pagos instantáneos. Puedes enviar un mensaje a Kafka cuando una transacción sea creada y luego consumir ese mensaje para continuar con el procesamiento de la transacción, como la actualización del saldo de la cuenta y la generación de un recibo.

1. **Integración entre microservicios**:

Kafka puede servir como un intermediario para la comunicación entre microservicios. En lugar de que los microservicios interactúen directamente entre sí, pueden intercambiar mensajes a través de Kafka, lo que permite desacoplar los servicios y mejorar la escalabilidad y la resiliencia.

**Ejemplo**:

Si un servicio de Transacciones necesita enviar una notificación al servicio de Clientes, en lugar de invocar directamente un método en ese servicio, el servicio de Transacciones podría publicar un mensaje en un tema de Kafka y el servicio de Clientes lo consumiría.

1. **APLICACIÓN EN EL PROYECTO DE MICROSERVICIOS**
   1. **Casos de uso real de Redis:**

* Caché de transacciones recientes: Al mantener las transacciones recientes de los clientes en Redis, puedes mostrar rápidamente el historial reciente de transacciones sin hacer una consulta costosa a la base de datos.
* Verificación rápida de saldos: Para operaciones como depósitos, retiros y transferencias, puedes consultar el saldo en Redis para validar la transacción antes de realizarla en la base de datos.

**Ejemplo:**

Si un cliente intenta hacer un retiro, el sistema primero consulta el saldo en Redis para validar si es suficiente para la transacción. Si no está en Redis, se realiza una consulta a la base de datos y el resultado se almacena en Redis.

* 1. **Casos de uso real de Kafka:**
* Envío de eventos de transacciones: Cada vez que un cliente realice una transacción (depósito, retiro, transferencia), podrías publicar un evento de transacción en un tema de Kafka, que puede ser consumido por otros microservicios para acciones como auditoría, notificación o análisis.

**Ejemplo:**

Cuando se realiza un depósito, el microservicio de transacciones emite un evento en Kafka, que es consumido por un servicio de auditoría para registrar la transacción, y por un servicio de notificaciones para enviar una confirmación al cliente.

* Sincronización de saldos: Si tienes múltiples microservicios que mantienen información sobre las cuentas, puedes usar Kafka para sincronizar los saldos entre ellos en tiempo real, asegurando que todos los microservicios tengan la misma información actualizada.

**Ejemplo:**

Cuando una transacción afecta el saldo de una cuenta, un mensaje de Kafka es publicado, y todos los microservicios que gestionan cuentas consumen el mensaje para actualizar sus registros de saldo.

1. **BENEFICIOS**

* Mejora del rendimiento: Redis reduce el tiempo de respuesta al almacenar datos en memoria, lo que aumenta la velocidad de acceso a la información crítica, como saldos y transacciones.
* Escalabilidad: Kafka permite escalar horizontalmente, procesando grandes volúmenes de eventos en tiempo real, y facilitando la comunicación asíncrona entre microservicios.
* Desacoplamiento de servicios: Ambos Redis y Kafka permiten desacoplar los microservicios, lo que mejora la resiliencia y facilita el mantenimiento y la extensión del sistema.

1. **CONCLUSIONES**

El uso de Redis y Kafka en un sistema bancario basado en microservicios proporciona mejoras significativas en términos de rendimiento, escalabilidad y resiliencia. Redis permite optimizar el acceso a datos frecuentes y gestionar sesiones de usuario, mientras que Kafka facilita la gestión de eventos y la comunicación asíncrona entre servicios. Ambas herramientas complementan perfectamente la arquitectura de microservicios, ofreciendo soluciones eficientes a problemas comunes en sistemas distribuidos.

Al integrar Redis y Kafka en tu sistema, se garantiza que el sistema bancario sea más rápido, eficiente, escalable y resiliente, permitiendo una mejor experiencia al usuario y un mejor manejo de grandes volúmenes de datos en tiempo real.